# Gym-ANM Tool

## 1. RL Environment

Deterministic environment

Deterministic environment - we know the outcome based on the current state. For instance, in a chess game, we know the exact outcome of moving any player.

Stochastic environment

stochastic environment - we cannot determine the outcome based on the current state. There will be a greater level of uncertainty. For example, we never know what number will show up when throwing a dice.

Fully observable environment

Fully observable - agent can determine the state of the system at all times. For example, in a chess game, the state of the system, that is, the position of all the players on the chess board, is available the whole time so the player can make an optimal decision.

## 2. Active Network Management (ANM)

ANM refers to the design of control schemes that modulate the generators, the loads, and/or the DES devices connected to the grid.

## 3. Gym-ANM Tool

gym-ANM is a framework **for designing reinforcement learning (RL) environments** that **model Active Network Management** (ANM) tasks in electricity distribution networks.

Gym-ANM environments do not solve ANM problems. it provides a programming interface to test and compare various optimization and RL algorithms that aim solve ANM problems.

| Pros | Cons |
|---|---|
| Good Documentation and repository with example | Need to define Network data. Design the env |
| Microgrid architecture design can change as we want. Possible to have complex systems with different types of energy sources | Energy sources are not divided, like solar or wind etc. |
| We can use environments which are design by someone else. | 1 slack generator + 1 slack bus are required in the network |
| environments generated by gym-ANM follow the OpenAI Gym framework. It is easy if already familiar with the openAI Gym framework. | |

Example of an environment built with the gym-ANM framework is the ANM6Easy-v0. It has its own rendering tool.

## 3.1 Installation

- Need python 3.7+
- Easy to install through pip or from source.

Issues found in installation

1. need to install urllib3 package

```
Python3 -m pip install –upgrade urllib3
```

## 3.2 TOOL

This tool divides the device set as load, generators, and storage.

Generator set divided in to 2 parts

- Slack generator
- non slack generators (renewable sources + other generators)

Renewable energy resources are counted as generators (inject power into the grid) without specifying the resource (Solar or wind -> all are considered as generators).

Exception is network should consist with exactly one slack generator and 1 slack bus – this is a requirement

- Slack bus - used to balance power flows in the network and provide a voltage reference. The slack bus can also either inject or withdraw power into/from the network, such that the total generation remains equal to the total load plus transmission losses, always.

- Slack generator - supplies as much real power and reactive power as needed for balancing the power flow considering power generation, load demand and losses in the system while keep the voltage constant.

## 3.3 Cost Calculation

For simplicity, this tool only considers 2 sources of costs

- energy losses
- violation of operational constraints

## 3.4 Usage of Gym-ANM

We can use Gym ANM tool to,

1. Design a new environment and do testing and comparing
2. Use existing env to train RL algorithm

### 3.4.1 Design new Environment

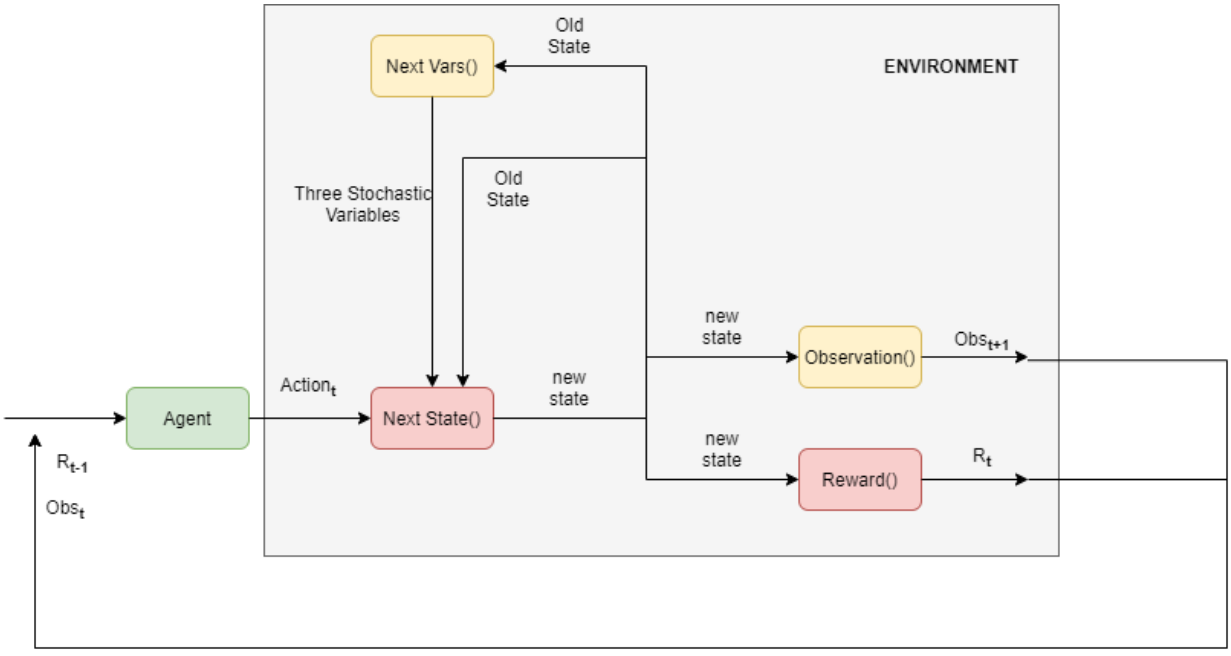


Figure 1: Gym ANM environment in high level

*Next_vars()* and observation() methods are the only component that are fully customizable when designing a new environment.

Next_state() and Reward() methods are built-in to the framework

1. New environments are created by implementing sub class of ANMEnv

2. Agent pass an action to the environment ($a_t$) in each time step

3. Next_var() function generate set of stochastic variables. internal variables model the temporal stochastic evolution of the electricity demand and of the maximum renewable energy production

4. Those stochastic variables and action use to simulate the distribution network and transition to a new state $S_{t+1}$

5. The next state() function behaves deterministically for a given network.
   All electrical quantities are updated by solving a set of network equations inside this method

   It first maps the selected action at to the current available action space A, before applying it to the environment. All the currents, voltages, energy storage levels, and power flows and injections are then updated, resulting in a new state st+1

6. Output observation vector and reward through the observation() and reward() methods.

7. Observation Space - set of default common observation spaces are provided by the framework. (We can use different observation vectors)

8. Reward calculation – Built in. we have no control over this method. Only initial parameter can change from our side

    a. Goal of reward function is to learn a control policy, that ensures a secure operation of the DN while minimizing its operating costs.
    b. Using clipping parameter, reward always keep in finite range.
    c. Reward calculations are based on,
        i. Total energy lost
        ii. Penalty associated with the violation of operating constants

iii. Weighting this penalty

9. Reach terminal state (Done = true). Then there is no solution to the power flow equations was found as a result of the action taken by the agent. This means that the power grid has collapsed and is often due to a voltage collapse problem.

Variables required to build new Gym-ANM environments.

Below variables are needed to create new Gym-ANM environment

We need to provide data related to a network (data related to loads, generators and power sources that are connected) this is describing the grid.

**Network** = {

'baseMVA' : Single integer value – represent the base power of the system (MVA). Use this to normalize the values to per unit

'bus' : Numpy 2D array – each row represent data (5 characteries) related to a single bus. Number of rows depend on the number of bus we need

'device' : Numpy 2D array – each row represent data (15 characteries) related to a single device. number of rows depend on the number of devices we need

'branch' : Numpy 2D array – each row represent data (8) related to a single branch. number of rows depend on the number of branches we need

}

**Observation** – 2 options

1. list of tuples of strings corresponding to the variables to include in observation vectors. (each tuple (x, y, z) refers to quantity x at nodes/devices/branches y, using units z)
2. Observation object can define as a customized function. It returns observation vectors. (pass state to the function)

Combinations for observation parameters are available.

**K** – number of auxiliary variables in the state vector

Number of auxiliary variable generated by the `next_vars()` function during the transition from timestep t to timestep t+1.

**Seed** – seed for random number generation - OPTIONAL

**Aux_bounds** – bounds for auxiliary variable - OPTIONAL

Hyperparameters

hyperparameter is a parameter that value is used to control the learning process.
Gym-ANM has hyperparameters defined. `Reward()` function relies on this parameters

1. **Lambda** - penalty weighting hyperparameter. The factor multiplying the penalty associated with violating operational constraints (used in reward)
2. **Gamma** – discount factor - the discount factor determining the weight of short tern rewards vs long-term rewards. discount factor [0, 1) is a fixed hyperparameter. (optimal policy must minimizes the expected sum of discounted costs)
3. **Time_t** – time interval between timesteps (fraction of hour)
4. **Clipping** – reward clipping parameter – to keep reward values in finite range - OPTIONAL

### Ways to get different ANM tasks

1. Change the topology and parameters of each device, bus, etc. of the network.
2. Changing next_var() – stochastic variables. It models the electricity demand of each load and maximum production at each generator could produce at time t. and auxiliary variables K.
3. Changing observation space using observation() method
4. Changing hyperparameters (reward relies on these parameters) – penalty weighting, delta t, clipping, discount factor.

### 3.4.2 Use existing Gym-ANM Environment

We can use existing Gym-ANM environment with RL agent implementation. Also we can customize the existing environments.

all Gym-ANM environments provide the same interface as traditional Gym environments.

**Existing Gym-ANM Environment – ANM6-Easy -** toy example

NOTE: We can customize this environment

This is 6 bus network with

- 1 transformer
- 3 loads
- 2 renewable energy sources
- 1 storage unit
- 1 fossil fuel generator used as slack generator
- The observation() component is the identity function in this environment. This leads to a fully observable environment with ot = st.

1. To limit the complexity of the task, the environment was designed to be fully deterministic: (we know the outcome based on the current state)
- Both the demand from loads and the maximum generation profiles from the renewable energies are modelled as fixed 24-hour time series that repeat every day, indefinitely.
- This is achieved by using a single auxiliary variable auxt(0) that represents the time of day and that can be used to index the fixed 24-hour time series.
  Note that the presence of auxt(0) makes the task an MDP, since future loads and generations can be expressed as functions of the time of day.
- These 24-hour time series were divided into 3 particular scenarios, each designed to highlight a specific challenge in ANM.

2. This environment has rendering tool. Which provides web interface. Currently only this environment has this rendering tool.

3. They also compared the performance of the soft actor-critic (SAC) and proximal policy optimization (PPO) RL algorithms against optimal MPC policy on this environment. with almost no hyperparameter tuning, the RL policies were already able to reach near-optimal performance.

4. They use MPC policy – it provides a loose lower bound on the best performance achievable in the environment. They consider 2 variants
    a. Constant policy – We can use it in any environment
    b. Perfect policy – only suitable for simple environments. Like ANM6-Easy.

What is MPC – Model predictive control

**MPC** uses a **model** of the plant to make predictions about future plant outputs.
Optimization strategy. strategy on a known model of the dynamics of the system, a multi-stage optimization problem is solved at each timestep over a finite time horizon.
The solution found is applied to the system at the current timestep, and the process is repeated at the next one, indefinitely.

## *References*


1. Read the docs - https://gym-anm.readthedocs.io/en/latest/
2. Research paper - https://arxiv.org/abs/2103.07932
3. Github – tool - https://github.com/robinhenry/gym-anm
4. Github – experiments in paper - https://github.com/robinhenry/gym-anm-exp
5. Github – Summer Internship 2021 implementations - https://github.com/anushaihalapathirana/RL-Gym-ANM-tool